Seguridad en Kubernetes

Luis Gutiérrez López

Índice

Manual de Seguridad en Kubernetes

Autenticación, Autorización, RBAC, Pod Security, Helm y Network Policies

Versión: 2026 Audiencia: Administradores de plataformas Kubernetes, SRE, DevOps y responsables de seguridad.

Compatibilidad: Kubernetes 1.29+ (válido para versiones modernas hasta 1.37 salvo cambios futuros menores).

---

1. Introducción

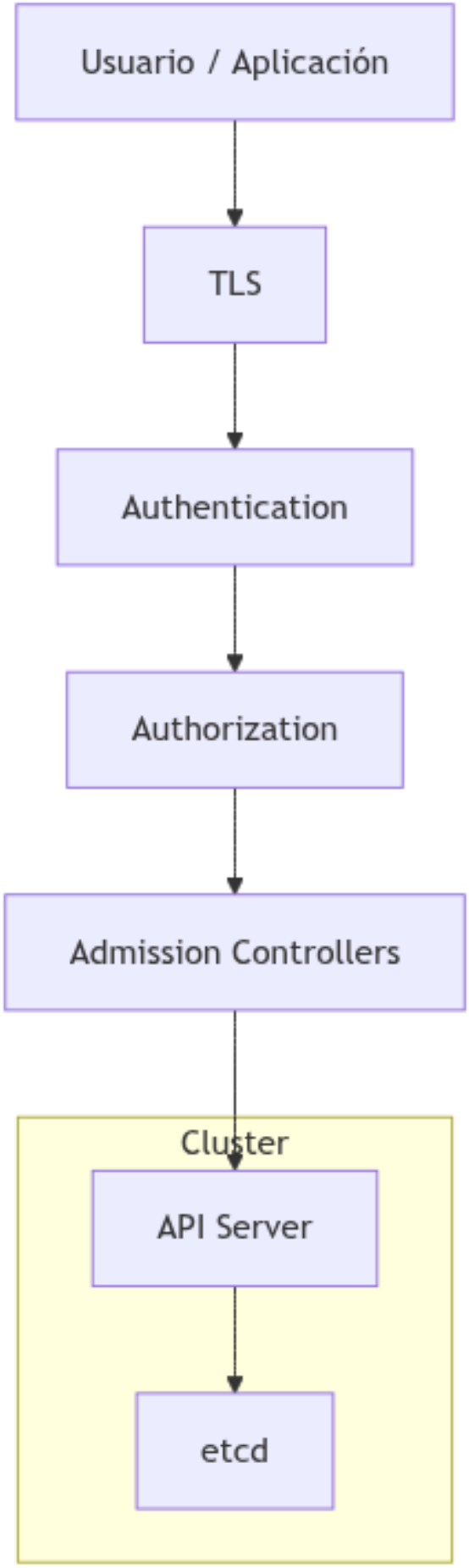
La seguridad en Kubernetes debe abordarse mediante una estrategia de defensa en profundidad.

Las principales áreas de protección son:

- Acceso al API Server.
- Gestión de identidades.
- Roles y permisos.
- Seguridad de workloads.
- Seguridad de red.
- Protección de imágenes.
- Auditoría y monitorización.
- Respuesta ante incidentes.

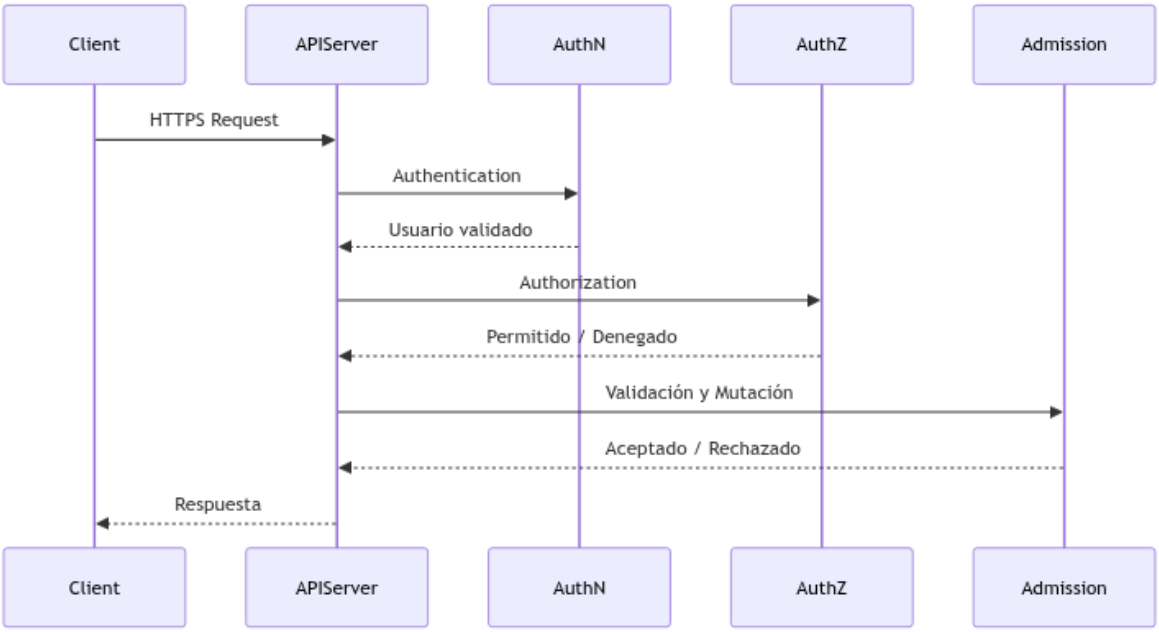
Un error común consiste en concentrar toda la seguridad en RBAC. Kubernetes requiere aplicar controles en todas las capas.

2. Arquitectura de Seguridad en Kubernetes



Cada petición que llega al API Server atraviesa varias fases antes de ser aceptada.

3. Flujo de Control de Acceso



Proceso:

1. Establecimiento TLS.
2. Autenticación.
3. Autorización.
4. Admission Control.
5. Persistencia en etcd.

4. Seguridad TLS

Objetivos

TLS proporciona:

- Confidencialidad.
- Integridad.
- Autenticación del servidor.
- Autenticación mutua opcional.

Kubernetes utiliza TLS para:

- kubectl <=> API Server
- API Server <=> kubelet
- API Server <=> etcd
- Comunicación interna del control plane

Verificación rápida

```
kubectl config view
```

Visualizar certificado:

```
openssl x509 -in apiserver.crt -text -noout
```

Recomendaciones * Utilizar certificados emitidos por CA corporativa. * Rotar certificados periódicamente. * Deshabilitar algoritmos obsoletos. * Revisar caducidades.

5. Authentication

Authentication responde a la pregunta:

> ¿Quién eres?

Kubernetes soporta múltiples mecanismos simultáneamente.

Métodos soportados

Certificados X.509

Método habitual para administradores.

```
CN=admin
O=platform-admins
```

OpenID Connect (OIDC)

Actualmente es el mecanismo recomendado para integrar:

- Keycloak
- Azure Entra ID
- Okta
- Google Identity

Service Accounts

Identidades utilizadas por los Pods.

Webhook Authentication

Delegación de autenticación hacia sistemas externos.

Nota técnica

Las contraseñas estáticas y los ficheros estáticos de usuarios deben considerarse obsoletos para entornos productivos modernos.

6. Authorization

Authorization responde a la pregunta:

> ¿Qué puede hacer este usuario?

Kubernetes soporta varios autorizadores. RBAC es el estándar actual.

Modos disponibles

- RBAC
- Node Authorizer
- Webhook Authorizer

ABAC continúa existiendo por compatibilidad histórica, pero se considera heredado frente a RBAC.

7. RBAC

Conceptos

RBAC controla permisos mediante:

- Role
- ClusterRole
- RoleBinding
- ClusterRoleBinding

Los permisos son acumulativos.

No existen reglas de tipo “deny”.

Modelo RBAC

Role

Permisos dentro de un namespace.

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role

metadata:
  name: pod-reader
  namespace: desarrollo

rules:
  - apiGroups: [""]
    resources:
      - pods
    verbs:
      - get
      - list
      - watch
```

RoleBinding

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding

metadata:
  name: pod-reader-binding
  namespace: desarrollo

subjects:
  - kind: User
    name: luis

roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

ClusterRole

Permisos globales o reutilizables.

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole

metadata:
  name: monitoring

rules:
  - apiGroups: [""]
    resources:
      - nodes
      - pods
    verbs:
```

```
    - get
    - list
```

Verificación de permisos

```
kubectl auth can-i get pods
```

Como otro usuario:

```
kubectl auth can-i list pods \
   --as=luis \
   -n desarrollo
```

Buenas prácticas RBAC

Correcto

Permisos mínimos.

```
verbs:
  - get
  - list
```

Evitar

```
verbs:
  - "*"
```

8. Admission Controllers

Los Admission Controllers actúan después de la autorización.

Pueden:

- Validar objetos.
- Modificar objetos.
- Añadir valores por defecto.
- Rechazar despliegues.

Ejemplos

- LimitRanger
- ResourceQuota
- MutatingAdmissionWebhook
- ValidatingAdmissionWebhook

Caso típico

Impedir imágenes no autorizadas.

```
registry.company.local/*
```

9. Pod Security Admission (PSA)

Cambio importante: Obsoleto

```
PodSecurityPolicy (PSP)
```

> PSP fue eliminado en Kubernetes 1.25.

Sustitución actual

```
Pod Security Admission (PSA)
```

Estable estable desde Kubernetes 1.25.

Niveles de seguridad

Privileged

Sin restricciones.

Baseline

Impide las elevaciones de privilegio más comunes.

Restricted

Configuración endurecida recomendada.

Habilitación mediante labels

```yaml
apiVersion: v1
kind: Namespace

metadata:
  name: producción
  labels:
    pod-security.kubernetes.io/enforce: restricted
    pod-security.kubernetes.io/audit: restricted
    pod-security.kubernetes.io/warn: restricted
```

10. Service Accounts

Cada Pod ejecuta una Service Account.

Crear una cuenta dedicada

```yaml
apiVersion: v1
kind: ServiceAccount

metadata:
  name: app-sa
```

Asociar a un Deployment

```yaml
spec:
  serviceAccountName: app-sa
```

Recomendación

No utilizar la Service Account default.

Asignar una específica por aplicación.

11. Helm y Seguridad

¿Qué es Helm?

Helm es el gestor de paquetes de Kubernetes.

Concepto	Descripción
Chart	Paquete Kubernetes
Release	Instancia desplegada
Repository	Repositorio de Charts

Estructura moderna de Chart

```
mychart
  |
  +--- Chart.yaml
  +--- values.yaml
  +--- values.schema.json
  +--- charts/
  +--- crds/
  +--- templates/
```

Instalación

```
helm version
```

Añadir repositorio:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
```

Instalación de aplicación

```
helm install nginx bitnami/nginx
```

Actualización

```
helm upgrade nginx bitnami/nginx
```

Rollback

```
helm rollback nginx 1
```

Advertencia

Helm 2 utilizaba Tiller.

Tiller fue eliminado completamente en Helm 3 y no debe aparecer en documentación moderna.

12. Seguridad de Infraestructura

La seguridad comienza fuera de Kubernetes.

Host Operating System

Recomendaciones:

- SO mínimo.
- Parcheado continuo.
- SELinux Enforcing.
- AppArmor cuando proceda.
- Acceso SSH restringido.

Seguridad del Cluster

- Rotación de certificados.
- Auditoría habilitada.
- Control de versiones.
- Restricción de acceso a etcd.

13. Seguridad de Imágenes

Reglas básicas

Utilizar imágenes mínimas

Preferible:

```
distroless
ubi-micro
alpine (cuando sea compatible)
```

Fijar versiones

Incorrecto:

```
image: nginx:latest
```

Correcto:

```
image: nginx:1.29.1
```

Firmado de imágenes

Herramientas habituales:

- Cosign
- Sigstore

14. Seguridad Runtime

Objetivo:

Detectar comportamientos anómalos en producción.

Eventos sospechosos

- Shell interactiva inesperada.
- Escalada de privilegios.
- Acceso a secretos.
- Conexiones salientes anómalas.

Falco

Proyecto CNCF ampliamente utilizado.

Ejemplo:

```
- rule: Terminal shell in container
  condition: >
    container and
    spawned_process and
    proc.name in (bash,sh,zsh)
```

15. Network Policies

Permiten controlar tráfico en capas L3 y L4.

Requisito importante

La CNI debe soportar NetworkPolicy.

Ejemplos:

- Calico
- Cilium
- Antrea

Si la CNI no las implementa, las políticas no tendrán efecto.

Política Default Deny

Recomendación inicial para entornos productivos.

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy

metadata:
  name: default-deny

spec:
  podSelector: {}

  policyTypes:
    - Ingress
    - Egress
```

Permitir acceso Frontend → Backend

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy

metadata:
  name: allow-frontend

spec:

  podSelector:
    matchLabels:
      role: backend

  ingress:
    - from:
      - podSelector:
          matchLabels:
            role: frontend
```

Permitir DNS

Frecuentemente olvidado en despliegues.

```yaml
egress:
- to:
  - namespaceSelector:
      matchLabels:
        kubernetes.io/metadata.name: kube-system

  ports:
  - protocol: UDP
    port: 53
```

Diagrama de Segmentación

16. Auditoría y Respuesta ante Incidentes

Habilitar Audit Logging

Ejemplo API Server:

```
--audit-log-path=/var/log/kubernetes/audit.log
```

Qué registrar

- Creación de usuarios.
- Cambios RBAC.
- Creación de Secrets.
- Eliminación de recursos.
- Escalados de privilegio.

17. Herramientas Recomendadas

Análisis de configuración

kube-bench

Verificación CIS.

kubescape

Análisis de seguridad.

kubeaudit

Auditoría de configuración.

Runtime

Falco

Detección de amenazas.

Supply Chain

Trivy

Escaneo de vulnerabilidades.

Grype

Escaneo de imágenes.

Cosign

Firma de artefactos.

18. Buenas Prácticas

Identidad

- Integrar OIDC corporativo.
- MFA obligatorio.
- Eliminar credenciales compartidas.

RBAC

- Aplicar mínimo privilegio.
- Revisiones periódicas.
- Evitar cluster-admin.

Networking

- Default Deny.
- Segmentación por namespaces.
- Cilium o Calico.

Workloads

- Ejecutar como usuario no root.
- readOnlyRootFilesystem.
- allowPrivilegeEscalation: false.

Supply Chain

- Firmar imágenes.
- Escanear vulnerabilidades.
- Utilizar repositorios privados.

19. Referencias Oficiales

Kubernetes

- Authentication https://kubernetes.io/docs/reference/access-authn-authz/authentication/
- RBAC https://kubernetes.io/docs/reference/access-authn-authz/rbac/
- Pod Security Admission https://kubernetes.io/docs/concepts/security/pod-security-admission/
- Pod Security Standards https://kubernetes.io/docs/concepts/security/pod-security-standards/
- Network Policies https://kubernetes.io/docs/concepts/services-networking/network-policies/

Helm

- Helm Documentation https://helm.sh/docs/
- Chart Structure https://helm.sh/docs/topics/charts/
- Helm RBAC https://helm.sh/docs/topics/rbac/
